Behavioral Cloning Project

Writeup

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

Rubric Points

Files Submitted

1. Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:

- model.py containing the script to create and train the model
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- writeup_report.pdf summarizing the results

2. Submission includes functional code

Using the Udacity provided simulator and drive.py file, the car can be driven autonomously around the track by executing

```
python drive.py model.h5
```

Note:

The speed setting in the 'drive.py' line 47 is changed to 25 (instead of 9) just to run the simulation faster.

3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain what the code does.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

The model selected for this project is based on the paper published from NVIDIA which can be found in this link (https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf). It is also presented in the course material. The model consists of a convolution neural networks with three 5x5 and two 3x3 filter sizes. The depths of the 5x5 convolutional layers are 24,36 and 48 with strides of 2x2 and for the 3x3 layers depth of 64 with stride of 3x3 are used.

The model uses '***relu'*** activations, and the data is normalized in the model using a Keras lambda layer (code line 116). In addition, the input image is cropped to avoid unnecessary part of the image data like horizon, trees and body of the car.

2. Attempts to reduce overfitting in the model

The model contains 50% dropout layers to reduce overfitting. It is used three times in the model (2 times in the convolutional layers and 1 time in the fully connected layer).

The model was trained and validated on data sets collected by driving the car in clockwise and counter clockwise directions. To add some randomness, during data collection there are times when the car is driven a little bit outside the road. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3. Model parameter tuning

The model used an '***adam'*** optimizer, so the learning rate was not tuned manually (model.py line 148).

4. Appropriate training data

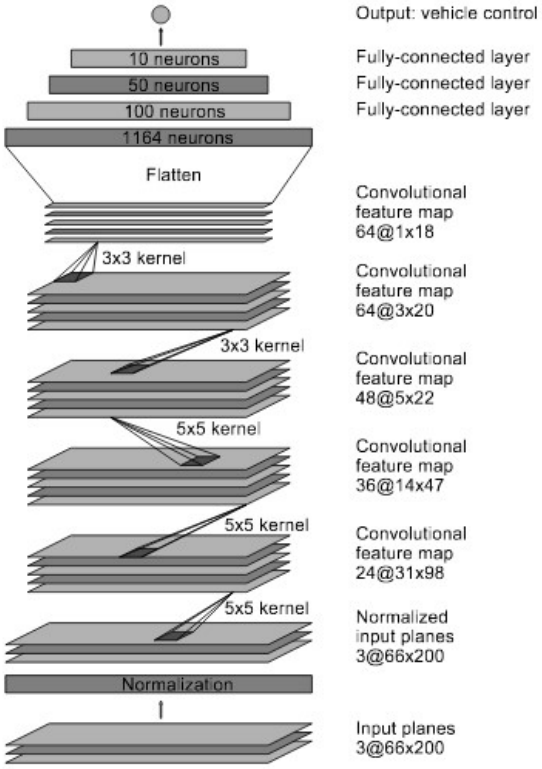
Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road by driving the car a little outside the road.

Model Architecture and Training Strategy

1. Solution Design Approach

My first step was to use a convolution neural network model based on the NVIDIA's end-to-end learning paper and the video demonstration provided in this course material. (see figure below)

Image and steering data is split into a training and validation set (80% vs 20%). To avoid/reduce effect of overfitting, 50% dropout layers are introduced in addition to the shuffling and different data sets used. I observed that the validation error is slightly higher than the training error although the error magnitude is very small (<0.03). The total data recorded is 28,026 (including the left and right camera images).



(**source**: https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf)

2. Creation of the Training Set & Training Process

When I recorded the data, most of the time I tried to drive the car keeping center of the lane and in other times the driving was intentionally a little outside the lane. To collect more data, driving is done clockwise and anticlockwise for at least 2 complete loops in each direction. A total of 28,026 image data is collected from the three cameras.

Here is an example image of center lane driving:


Images showing center lane driving when the car is slightly off the road.


Images from left and right cameras when center lane driving are shown below.


I used this all those data for training the model. The images from left and right cameras are used as if the images are coming from the center camera by adding a correction factor on the steering angle information. The correction factor, which can be considered as one hyper parameter, is adjusted by trial and error. A number in between 0.18 to 0.23 can be used (on average 0.2) as correction factor and produces very good driving result test track 1.

Output

- output.mp4 → video created by 'video.py' script
- screenRecorded.mp4 → Screen recorded video